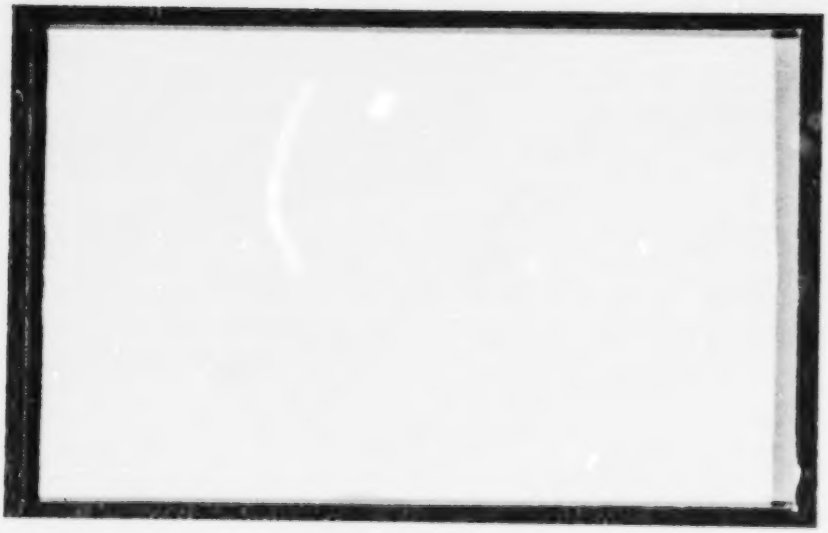
UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

ETUDE ET EXTENSION D'UN ALGORITHME DE MURCHLAND

pour corriger la matrice des longueurs des plus courts chemins après modification de la longueur d'un seul arc dans un réseau

par

René Dionne

Université de Montréal
Publication # 122 - Département d'informatique
Publication # 20 - Centre de recherche sur les transports

Révisé octobre 1974

Dépôt légal - 1er trimestre 1978

RÉSUMÉ

Un algorithme publié par Murchland (1970) permet de recalculer de façon économique la matrice des longueurs des plus courts chemins après modification de la longueur d'un seul arc dans un réseau. Nous étudions cet algorithme du point de vue de la théorie et du point de vue des calculs. Nous suggérons une légère modification de l'algorithme pour en améliorer l'efficacité. Murchland traite seulement le cas orienté ($d_{ij} \neq d_{ji}$, où d_{ij} représente la longueur de l'arc qui va du sommet i au sommet j). Nous traitons en plus le cas symétrique ($d_{ij} = d_{ji}$) .

ABSTRACT

Murchland (1970) published an algorithm for recomputing all shortest paths after a change in the length of only one arc in a network. We study that algorithm both from a theoretical and a computational point of view. We suggest a slight modification of the algorithm to improve computational efficiency. Murchland deals with the oriented case ($d_{ij} \neq d_{ji}$, where d_{ij} = length of arc from i to j). We also study the non-oriented case ($d_{ij} = d_{ji}$).

Introduction

En traitant le problème du choix optimal d'un réseau (cf. Stairs (1968), par exemple), on se trouve souvent dans la situation suivante: ayant modifié la longueur d'un seul arc d'un réseau, on aimerait connaître la nouvelle matrice des longueurs des plus courts chemins sans la recalculer au complet par l'algorithme de Floyd-Murchland (c'est-à-dire l'algorithme défini par Floyd (1962), puis redécouvert et exploité par Murchland; pour plus de détails, cf. Dreyfus (1969)).

Murchland (1970) a proposé un algorithme pour calculer cette nouvelle matrice des longueurs des plus courts chemins. Le présent rapport rend compte d'une étude qui a été menée dans le but de connaître l'efficacité de ce dernier algorithme (Murchland (1970), p. 39 et p. 43). Il présente d'abord l'algorithme pour le cas orienté, selon un point de vue qui a permis d'apporter une amélioration à l'algorithme tel que défini par Murchland. Il traite ensuite le cas non orienté, ce qui n'a pas été fait par Murchland. Pour chacun de ces deux cas, sont présentés des résultats numériques obtenus à l'aide de l'ordinateur CDC CYBER 74 du Centre de Calcul de l'Université de Montréal.

1. Le cas orienté

Le problème à résoudre s'énonce comme suit. Pour un réseau donné, on connaît la longueur de tous les plus courts chemins entre chaque paire de points. On modifie la longueur d'un arc (ou encore on ajoute ou on supprime un arc). On doit calculer les nouvelles longueurs des plus courts chemins, mais on serait mal avisé de recalculer les longueurs de tous les plus courts chemins. On doit définir un algorithme qui permette de repérer les plus courts chemins qui sont effectivement modifiés et qui ne refait le calcul que pour ceux-ci. Il faut bien entendu garder le nombre d'opérations à un minimum.

Le réseau est défini par la donnée d'un graphe $G = (N,A)$ et d'une matrice D.

$N = \{1,2,\ldots,n\}$ = ensemble des points ou sommets du graphe G

A = ensemble des arcs du graphe G

$D = (d_{ij})_{nxn}$, où d_{ij} = longueur de l'arc du sommet i au sommet j

d'_{ij} = valeur modifiée de d_{ij} .

Lorsqu'il n'y a pas d'arc de i à j, on conviendra de poser $d_{ij} = M$ (M grand; par exemple, $M = n \cdot \max_{(i,j)\in A} d_{ij}$). Ainsi on pourra assimiler le cas où un arc est ajouté ($d'_{ij} < d_{ij} = M$) au cas où la longueur d'un arc est simplement réduite ($d'_{ij} < d_{ij} < M$). On traitera la suppression d'un arc ($d_{ij} < d'_{ij} = M$) de la même façon que son simple allongement ($d_{ij} < d'_{ij} < M$).

On connaît (ou on peut obtenir par l'algorithme de Floyd-Murchland) la matrice des plus courts chemins $C = (c_{ij})_{nxn}$, où c_{ij} désigne la longueur du plus court chemin de i à j. A noter que $c_{ij} \geq M$ signifie qu'il n'y a pas de chemin de i à j.

On supposera aussi que $d_{ii} = c_{ii} = 0$, $i \in N$

et $d_{ij} > 0$, $i \neq j$.

Enfin, rappelons que le réseau étudié ici est orienté (d_{ij} peut différer de d_{ji}).

1.1 Diminution de la longueur d'un arc

L'arc modifié est (j,k) et sa nouvelle longueur, d'_{jk}.

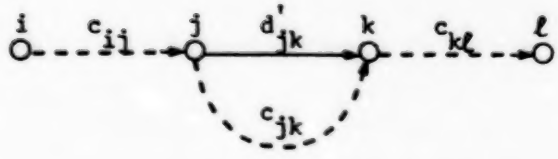
$$0 < d'_{jk} < d_{jk} \leq M .$$

Si $c_{jk} \leq d'_{jk}$, il n'y a pas de changement dans C. On supposera donc ici que

$$d'_{jk} < c_{jk} \leq d_{jk} .$$

On doit déterminer toutes les paires (i,ℓ) de sommets pour lesquelles $c_{i\ell}$ doit être modifié (nous parlerons plus simplement des paires (i,ℓ) à modifier). Avant la modification de la longueur de l'arc (j,k), deux cas sont possibles pour chaque paire (i,ℓ) : ou bien le plus court chemin de i à ℓ passe par l'arc (j,k) (1er cas) ou bien il n'y passe pas (2e cas).

1er cas



——→ représente un arc

- - - - - → représente un plus court chemin

Le plus court chemin de i à ℓ passe par l'arc (j,k) et se décompose en trois parties:

1) le plus court chemin de i à j, de longueur c_{ij}

2) le plus court chemin de j à k, de longueur d_{jk}

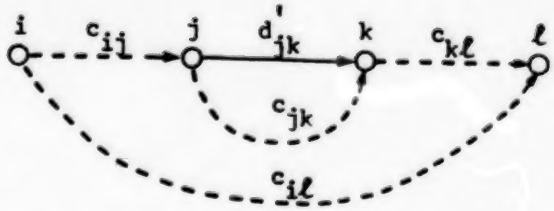
3) le plus court chemin de k à ℓ, de longueur $c_{k\ell}$.

$c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} > c_{ij} + d'_{jk} + c_{k\ell}$.

Il est clair que $c_{i\ell}$ doit être modifié comme suit:

$$c_{i\ell} := c_{ij} + d'_{jk} + c_{k\ell} .$$

2e cas



Le plus court chemin de i à ℓ ne passe pas par l'arc (j,k) .

$$c_{i\ell} < c_{ij} + d_{jk} + c_{k\ell} \ .$$

Il se peut qu'il faille tout de même modifier $c_{i\ell}$. En effet, si $c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$, un nouveau plus court chemin se trouve créé par la modification de l'arc (j,k) , qui va de i à ℓ en passant par l'arc (j,k) . Dans ce cas, il faut modifier $c_{i\ell}$ comme suit:

$$c_{i\ell} := c_{ij} + d'_{jk} + c_{k\ell} \ .$$

D'après la discussion qui précède, on peut énoncer la proposition 1 .

Proposition 1:

$c_{i\ell}$ doit être modifié $\Longleftrightarrow c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$.

La proposition 1 permet de repérer toutes les paires (i,ℓ) à modifier et seulement elles. Cependant, l'utilisation de la proposition 1 implique l'énumération de $n(n-1)$ paires pour leur faire subir le test "est-ce que $c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$? ". On aura moins de paires (i,ℓ) à énumérer si l'on parvient au préalable à éliminer à l'aide d'un test simple un certain nombre d'indices i , $i \in N$, dont on est certain qu'ils ne peuvent former le premier élément d'une paire (i,ℓ) à modifier. Soit N_1 l'ensemble des indices i qui ne sont pas éliminés ainsi. On peut procéder de façon analogue pour les indices ℓ , $\ell \in N$, et former N_2 , l'ensemble des indices ℓ qui peuvent être le deuxième élément d'une paire (i,ℓ) à modifier.

$N_1 \subset N$

$N_1 = \{ i \mid i$ est éligible à former le premier élément d'une paire (i,ℓ) à modifier $\}$

$N_2 \subset N$

$N_2 = \{ \ell \mid \ell$ peut former le deuxième élément d'une paire (i,ℓ) à modifier $\}$

Précisons maintenant la nature des tests qui définissent N_1 et N_2 .

Proposition 2:

$c_{ik} < c_{ij} + d'_{jk} \Longrightarrow i$ ne peut former le premier élément d'une paire (i,ℓ) à modifier .

Preuve: $c_{ik} \leq c_{ij} + d'_{jk}$

$\forall \ell$, $c_{ik} + c_{k\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$

Or $c_{i\ell} \leq c_{ik} + c_{k\ell}$ (propriété des plus courts chemins)

$\Longrightarrow \forall \ell$, $c_{i\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$

$\Longleftrightarrow$ $c_{i\ell}$ ne peut être modifié (par la proposition 1) .

Proposition 3:

$c_{j\ell} < d'_{jk} + c_{k\ell} \Longrightarrow \ell$ ne peut former le deuxième élément d'une paire (i,ℓ) à modifier .

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} < c_{ij} + d'_{jk} + c_{k\ell}$

Fin de la preuve.

Nous définissons donc:

$$N_1 = \{ i \mid c_{ik} > c_{ij} + d'_{jk} \}$$

$$N_2 = \{ \ell \mid c_{j\ell} > d'_{jk} + c_{k\ell} \}$$

$|N_1| = n_1$ et $1 < n_1 < n$

$|N_2| = n_2$ et $1 < n_2 < n$

Les $n_1\, n_2$ paires $(i,\ell) \ni i \in N_1$ et $\ell \in N_2$ sont les seules paires susceptibles d'être modifiées. La définition de N_1 et de N_2 mène directement à l'algorithme D (D pour diminution de la longueur d'un arc).

L'algorithme D

(Si $c_{jk} \leq d'_{jk}$, il n'y a pas de changements dans C .)

d1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

d2) $\forall i \in N$, vérifier si $c_{ik} > c_{ij} + d'_{jk}$.
Si oui, $N_1 := N_1 \cup \{i\}$ et $c_{ik} := c_{ij} + d'_{jk}$.

d3) $\forall \ell \in N$, vérifier si $c_{j\ell} > d'_{jk} + c_{k\ell}$.
Si oui, $N_2 := N_2 \cup \{\ell\}$ et $c_{j\ell} := d'_{jk} + c_{k\ell}$.

d4) $\forall i \in N_1$ et $\forall \ell \in N_2$, vérifier si $c_{i\ell} > c_{ij} + c_{j\ell}$.
(à noter que depuis l'étape d3, $c_{j\ell} = d'_{jk} + c_{k\ell}$)
Si oui, $c_{i\ell} := c_{ij} + c_{j\ell}$.

Cet algorithme est essentiellement le même que celui proposé par Murchland (1970, page 39).

Nombre d'opérations élémentaires

L'utilisation de l'algorithme de Floyd-Murchland pour obtenir la nouvelle matrice C exige de l'ordre de $2n^3$ opérations élémentaires (n^3 additions, n^3 comparaisons). L'algorithme D ne demande que $2(2n + n_1 n_2)$ opérations élémentaires:

$2n$ à l'étape d2 (n additions et n comparaisons)

$+ 2n$ à l'étape d3

$+ 2n_1n_2$ à l'étape d4 .

Dans un grand réseau, n_1 et n_2 devraient être considérablement plus petits que n . Le contraire signifierait que tous les plus courts chemins ont tendance à passer par le même arc (j,k) .

$$n_1 << n \qquad \text{et} \qquad n_2 << n \ .$$

Afin d'obtenir un estimé grossier du rapport du temps de calcul pour l'algorithme D au temps de calcul pour l'algorithme de Floyd-Murchland, considérons le rapport R du nombre d'opérations élémentaires exigées par l'algorithme D au nombre d'opérations élémentaires exigées par l'algorithme de Floyd-Murchland. Pour n assez grand, on a:

$$R = \frac{2(2n + n_1n_2)}{2n^3} = \frac{2}{n^2} + \frac{n_1n_2}{n^3} \cong \frac{n_1n_2}{n^3} << \frac{n^2}{n^3} = \frac{1}{n} \ .$$

Résultats expérimentaux

Pour recueillir de l'information sur le comportement numérique de l'algorithme D, nous avons résolu 47 problèmes. Ces problèmes sont constitués à partir de quatre réseaux; pour chacun de ces réseaux, un certain nombre (que nous appellerons NPROB) de problèmes a été obtenu en modifiant (diminuant) successivement la longueur de NPROB arcs, chaque modification donnant lieu à un problème. Nous ne jugeons pas opportun de donner ici la description complète de toutes ces données numériques, mais nous les gardons disponibles pour toute demande éventuelle de consultation. Et nous nous intéressons ici aux résultats numériques obtenus.

Pour chacun des problèmes traités, nous avons recueilli les valeurs de n_1 et n_2, de NMOD (i.e. le nombre d'éléments de la matrice C effectivement modifiés) et de T (i.e. le temps de calcul en secondes pour l'algorithme D).

On trouvera les résultats numériques dans le tableau 1, où nous avons dû les présenter de façon condensée (sous forme de moyennes) pour éviter de provoquer une avalanche de valeurs numériques à cause du grand nombre de problèmes traités (47). Dans le tableau 1, lorsqu'on lit la valeur de n_1 pour un réseau donné, il s'agit en fait de la valeur moyenne de n_1 pour les NPROB problèmes rattachés à ce réseau. De même pour n_2, NMOD et T. Pour fins de comparaison, les mêmes problèmes ont été traités par l'algorithme de Floyd-Murchland et le tableau 1 donne aussi le temps de calcul moyen en secondes pour cet algorithme (T_{floyd}).

Dans le tableau 1, il est particulièrement intéressant de noter comment le rapport T/T_{floyd} a tendance à décroître rapidement quand n croît, ce qui correspond grosso modo à l'estimé que nous avons fait précédemment en considérant le rapport R. Cependant, il ne s'agit là que d'une tendance, car, pour un problème donné, les valeurs de n_1 et de n_2 (et donc de T) dépendent non seulement de n, mais aussi de la structure du réseau et de la position stratégique de l'arc modifié.

	n	NPROB	n_1	n_2	NMOD	T	T_{floyd}	T/T_{floyd}
réseau # 1	22	10	9	6	22	.0013	0.07	1.9%
réseau # 2	35	10	15	14	95	.0022	0.29	0.8%
réseau # 3	50	15	29	20	157	.0080	0.82	1.0%
réseau # 4	70	12	34	28	266	.0090	2.22	0.4%

TABLEAU 1

1.2 Augmentation de la longueur d'un arc

Pour les fins de la discussion qui suit, excluons le cas où $c_{jk} < d_{jk}$, car alors C reste inchangée. On a donc:

$$0 < c_{jk} = d_{jk} < d'_{jk} \leq M .$$

Proposition 4:

$c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \Longrightarrow$ on ne doit pas modifier $c_{i\ell}$.

Preuve:

$c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \Longrightarrow$ le plus court chemin de i à ℓ ne passe pas par l'arc (j,k)

$\Longrightarrow$ il ne faut pas modifier $c_{i\ell}$

Fin de la preuve.

On ne conservera donc que des paires (i,ℓ) pour lesquelles l'égalité $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$ a lieu, et on rejettera celles pour lesquelles l'égalité n'a pas lieu. Cependant, il est important de noter que même si l'égalité a lieu, il se peut qu'on n'ait pas à modifier $c_{i\ell}$. Cela se produit lorsqu'il existe plus d'un chemin de longueur $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$ pour aller de i à ℓ. Dans ce cas, en allongeant l'arc (j,k), on allonge aussi le plus court chemin passant par (j,k) et allant de i à ℓ, mais il existe un autre chemin de longueur $c_{i\ell}$, qui, lui, n'est pas modifié.

Lorsqu'on diminue la longueur d'un arc, le plus court chemin entre deux sommets quelconques peut tout au plus diminuer en longueur: son

parcours ne change pas. Mais, si l'on allonge un arc, le parcours du plus court chemin entre deux sommets peut changer complètement. C'est pourquoi il faut faire un calcul de plus court chemin pour toute paire (i,ℓ) susceptible d'être modifiée, c'est-à-dire pour toute paire (i,ℓ) $\ni c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$.

Proposition 5:

$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow$ i ne peut former le premier élément d'une paire (i,ℓ) à modifier .

Preuve:

$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow c_{ik} < c_{ij} + d_{jk}$

$\Longrightarrow \quad \forall \ell \ , \ c_{ik} + c_{k\ell} < c_{ij} + d_{jk} + c_{k\ell}$

Or $c_{i\ell} \leq c_{ik} + c_{k\ell}$ (propriété des plus courts chemins) .

Donc $\forall \ell \ , \ c_{i\ell} < c_{ij} + d_{jk} + c_{k\ell}$

et, par la proposition 4, $c_{i\ell}$ ne peut être modifié.

Proposition 6:

$c_{j\ell} \neq d_{jk} + c_{k\ell} \Longrightarrow \ell$ ne peut former le deuxième élément d'une paire (i,ℓ) à modifier .

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} < c_{ij} + d_{jk} + c_{k\ell}$.

Une application judicieuse des propositions 4, 5 et 6 mène directement à l'algorithme A (A pour augmentation). Cette fois-ci, on

définit:

$$N_1 = \{i \mid c_{ik} = c_{ij} + d_{jk}\}$$

$$N_2 = \{\ell \mid c_{j\ell} = d_{jk} + c_{k\ell}\}$$

Il est important de remarquer que

$$(i,\ell) \in N_1 \times N_2 \not\Longrightarrow c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} .$$

Avant d'entreprendre le calcul du plus court chemin pour une paire $(i,\ell) \in N_1 \times N_2$, on vérifiera si cette paire est susceptible d'être modifiée, c'est-à-dire si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$ (cf. proposition 4).

L'algorithme A

(Si $c_{jk} < d_{jk}$, la matrice C n'est pas modifiée.)

a1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

a2) $\forall i \in N$, si $c_{ik} = c_{ij} + d_{jk}$, $N_1 := N_1 \cup \{i\}$.

a3) $\forall \ell \in N$, si $c_{j\ell} = d_{jk} + c_{k\ell}$, $N_2 := N_2 \cup \{\ell\}$.

a4) $\forall$ paire $(i,\ell) \in N_1 \times N_2$, si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$, on conserve la paire (i,ℓ) et $c_{i\ell} := d_{i\ell}$.

(remarque: $d_{i\ell}$ est la plus petite valeur que l'on puisse attribuer a priori à $c_{i\ell}$, dont la véritable valeur sera calculée à l'étape a5.)

a5) Appliquer l'algorithme de Floyd-Murchland à toutes les paires (i,ℓ) conservées à l'étape a4, i.e.

pour $m := 1$ **jusqu'à** n

et pour chaque paire (i,ℓ) conservée à l'étape a4

si $c_{im} + c_{m\ell} < c_{i\ell}$ **alors** $c_{i\ell} := c_{im} + c_{m\ell}$.

Cet algorithme diffère essentiellement de celui proposé par Murchland (1970 , page 43): l'étape a4 n'existe pas dans l'algorithme de Murchland. En ajoutant l'étape a4, on se trouve à éviter de faire l'onéreux calcul du plus court chemin pour toute paire $(i,\ell) \in N_1 \times N_2$ pour laquelle $c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell}$. On diminue ainsi le temps de calcul, ainsi que le confirment les résultats numériques que nous avons recueillis (cf. tableau 2).

Nombre d'opérations élémentaires

Supposons qu'à l'étape a4, on conserve n_{12} paires $(i,\ell) \in N_1 \times N_2$.

$$n_{12} \leq n_1 n_2$$

Alors le nombre d'opérations élémentaires pour l'algorithme A s'évalue comme suit:

	nombre d'additions	nombre de comparaisons
à l'étape a2,	n	n
à l'étape a3,	n	n
à l'étape a4,	$n_1 n_2 + n_1$ (ou $n_1 n_2 + n_2$)	$n_1 n_2$
à l'étape a5,	$n_{12} n$	$n_{12} n$
au total:	$2n + n_1 + n_1 n_2 + n_{12} n$	$2n + n_1 n_2 + n_{12} n$

Négligeant le terme n_1 qui a peu d'importance par rapport aux autres, on obtient:

$$\begin{aligned}
\text{nombre d'opérations élémentaires} &= 2(2n + n_1n_2 + n_{12}n) \\
&\leq 2(2n + n_1n_2 + n_1n_2n) \\
&= 2((2 + n_1n_2)n + n_1n_2) \\
&\cong 2(n_1n_2n + n_1n_2) \\
&= 2(n_1n_2(n + 1)) \\
&\cong 2n_1n_2n \quad , \text{ pour } n \text{ assez grand} \\
&<< 2n^3 \text{ , puisque } n_1 << n \text{ et } n_2 << n .
\end{aligned}$$

$$R \cong \frac{2n_1n_2n}{2n^3} = \frac{n_1n_2}{n^2} << 1$$

Le cas de l'allongement d'un arc se montre plus lourd à traiter que le cas du raccourcissement. Dans ce dernier cas, s'il faut modifier $c_{i\ell}$, il suffit de changer sa valeur ($c_{i\ell} := c_{ij} + d_{jk} + c_{k\ell}$). Dans le cas de l'allongement, s'il faut modifier $c_{i\ell}$, il faut faire un calcul de plus court chemin (étape a5). On ne peut espérer réduire le nombre d'opérations élémentaires qu'exige l'étape a5, puisque l'algorithme de Floyd-Murchland utilisé effectue ce calcul en un nombre minimum d'opérations (Murchland (1970), page 13, théorème 2).

Résultats expérimentaux

Les résultats numériques pour l'algorithme A apparaissent dans le tableau 2. Ce tableau s'interprète de la même façon que le tableau 1. On y trouve cependant de nouvelles valeurs, n_{12} et T'. T' représente le temps de calcul en secondes pour l'algorithme A sans l'étape a4. T' est donc le temps de calcul qu'on obtient si l'on effectue un calcul de plus court chemin pour toutes les paires $(i,\ell) \in N_1 \times N_2$, ce qui correspond à l'algorithme proposé par Murchland (1970, page 43). Dans le tableau 2, on remarquera que $n_{12} < n_1 n_2$ et donc que $(n_1 n_2 - n_{12})$ représente un nombre appréciable de paires (i,ℓ) pour lesquelles le calcul du plus court chemin est supprimé à l'aide de l'étape a4; cela se reflète dans le fait que $T < T'$. Finalement, on remarquera qu'en utilisant l'algorithme A plutôt que l'algorithme de Floyd-Murchland, on réalise une nette économie en temps de calcul $(T < T_{floyd})$. De plus, le rapport T/T_{floyd} suit une tendance à la décroissance quand n croît (exception faite pour le réseau # 3, qui engendre une valeur de T anormalement élevée).

	n	NPROB	n_1	n_2	n_{12}	NMOD	T	T'	T_{floyd}	T/T_{floyd}
réseau # 1	22	10	11	7	38	36	.010	.022	.07	14%
réseau # 2	35	10	18	22	73	71	.030	.142	.29	10%
réseau # 3	50	15	27	28	206	168	.113	.493	.82	14%
réseau # 4	70	12	26	30	162	158	.121	.412	2.22	5%

TABLEAU 2

2. Le cas symétrique

Le problème à traiter reste le même que dans le cas orienté, sauf pour la symétrie (le réseau est maintenant symétrique: $d_{ij} = d_{ij}$, $\forall i$ et $\forall j$) et pour ses conséquences immédiates (par exemple, $c_{ij} = c_{ji}$, $\forall i$ et $\forall j$) . Dans le cas symétrique, on ne parle pas d'arcs, mais d'arêtes, et on ne parle pas de chemins, mais de chaînes. Pour tout éclaircissement sur cette terminologie, nous renvoyons le lecteur au livre de Berge (1967).

Nous proposons ici deux algorithmes pour calculer la nouvelle matrice des longueurs des plus courtes chaînes après modification d'une seule arête (j,k) . Le premier algorithme s'applique dans le cas d'une diminution de la longueur de l'arête (j,k) ; le second, dans le cas d'une augmentation. Pour obtenir ces algorithmes, nous avons procédé de la même façon que dans le cas orienté, en faisant les transpositions dictées par la symétrie. C'est pourquoi nous serons brefs dans l'exposé qui suit.

A notre connaissance, seuls Boyce *et al.* (1973) ont proposé des algorithmes pour le problème étudié ici (cas symétrique). Nous mentionnerons au passage en quoi leurs algorithmes ressemblent aux nôtres ou en diffèrent.

2.1 Diminution de la longueur d'une arête

Soit donc (j,k) l'arête modifiée et d'_{jk} , sa nouvelle longueur. Si $c_{jk} \leq d'_{jk}$, il n'y a pas de changement dans C . On supposera donc

que

$$0 < d'_{jk} < c_{jk} \leq d_{jk} \leq M .$$

Posons P = l'ensemble des paires (i,ℓ) à modifier

$$E = \{ (i,\ell) \mid c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell} \}$$

$$E^S = \{ (i,\ell) \mid (\ell,i) \in E \} = \text{le symétrique de } E .$$

Proposition 7: $P = E \cup E^S$.

Preuve: Soit $(i,\ell) \in P$.

Il faut donc modifier $c_{i\ell}$ et cela implique qu'on peut construire une plus courte chaîne qui emprunte l'arête (j,k) et qui est plus courte que $c_{i\ell}$. Puisque l'arête (j,k) ne comporte aucune orientation, cette plus courte chaîne est formée d'une des deux façons suivantes:

1) elle emprunte la plus courte chaîne entre i et j ,
l'arête (j,k) entre j et k ,
et la plus courte chaîne entre k et ℓ .
Dans ce cas, $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} > c_{ij} + d'_{jk} + c_{k\ell}$.

2) elle emprunte la plus courte chaîne entre i et k ,
l'arête (j,k) entre k et j ,
et la plus courte chaîne entre j et ℓ .
Dans ce cas, $c_{i\ell} = c_{ik} + d_{jk} + c_{j\ell} > c_{ik} + d'_{jk} + c_{j\ell}$.

On a donc:

$$(i,\ell) \in P \Longrightarrow \begin{cases} c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell} \\ \underline{\text{ou}} \quad c_{i\ell} > c_{ik} + d'_{jk} + c_{j\ell} \end{cases}$$

A cause de la symétrie,

$$c_{i\ell} > c_{ik} + d'_{jk} + c_{j\ell} \Longleftrightarrow c_{\ell i} > c_{\ell j} + d'_{jk} + c_{ki}$$

Et on peut réécrire:

$$(i,\ell) \in P \Longrightarrow (i,\ell) \in E \ \underline{\text{ou}} \ (\ell,i) \in E \Longleftrightarrow (i,\ell) \in E \cup E^s$$

Donc $P \subseteq E \cup E^s$.

Par ailleurs, $E \cup E^s \subseteq P$.

En effet, $(i,\ell) \in E \Longleftrightarrow c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$

$\Longrightarrow c_{i\ell}$ à modifier

$\Longleftrightarrow (i,\ell) \in P$

De même, $(i,\ell) \in E^s \Longleftrightarrow (\ell,i) \in E$

$\Longleftrightarrow c_{\ell i} > c_{\ell j} + d'_{jk} + c_{ki}$

$\Longrightarrow c_{\ell i}$ à modifier

$\Longrightarrow c_{i\ell}$ à modifier

(puisqu'à cause de la symétrie, on ne peut modifier $c_{\ell i}$ sans modifier $c_{i\ell}$)

$\Longleftrightarrow (i,\ell) \in P$

Corollaire 7a: Pour connaître l'ensemble des paires (i,ℓ) à modifier, il suffit de considérer l'ensemble

$$E = \{ (i,\ell) \mid c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell} \} ,$$

puisque $c_{i\ell}$ et $c_{\ell i}$ doivent être toujours modifiés simultanément.

Proposition 8:

$c_{ik} \leq c_{ij} + d'_{jk} \Longrightarrow$ i ne peut constituer le premier élément d'une paire (i,ℓ) à modifier.

Preuve: Comme dans le cas orienté (voir la preuve de la proposition 2), on a:

$$\forall \ell \ , \ c_{i\ell} \leq c_{ik} + c_{k\ell} \leq c_{ij} + d'_{jk} + c_{k\ell} \ .$$

Et par le corollaire 7a, on n'a pas à considérer l'indice i .

Proposition 9:

$c_{j\ell} \leq d'_{jk} + c_{k\ell} \Longrightarrow \ell$ ne peut constituer le deuxième élément d'une paire (i,ℓ) à modifier.

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$.

L'utilisation judicieuse du corollaire 7a et des propositions 8 et 9 nous donne l'algorithme DS (diminution, symétrique).

L'algorithme DS

(Si $c_{jk} \leq d'_{jk}$, la matrice C reste inchangée.)

ds1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

ds2) $\forall\, i \in N$, si $c_{ik} > c_{ij} + d'_{jk}$,

$$N_1 := N_1 \cup \{i\} \ , \quad c_{ik} := c_{ij} + d'_{jk} \quad \text{et} \quad c_{ki} := c_{ik} \ .$$

ds3) $\forall\, \ell \in N$, si $c_{j\ell} > d'_{jk} + c_{k\ell}$,

$$N_2 := N_2 \cup \{\ell\} \ , \quad c_{j\ell} := d'_{jk} + c_{k\ell} \quad \text{et} \quad c_{\ell j} := c_{j\ell} \ .$$

ds4) $\forall\, (i,\ell) \in N_1 \times N_2$, si $c_{i\ell} > c_{ij} + c_{j\ell}$, $c_{i\ell} := c_{ij} + c_{j\ell}$

$$\text{et} \quad c_{\ell i} := c_{i\ell} \ .$$

Hormis quelques légères différences au niveau des détails, l'algorithme DS et l'algorithme "link addition" proposé par Boyce *et al.* (1973, p. 524) sont essentiellement le même algorithme.

Nombre d'opérations élémentaires

Comme le montre une simple comparaison des algorithmes D et DS, l'algorithme DS exige le même nombre d'opérations élémentaires que l'algorithme D, soit $2(2n + n_1 n_2)$. Cependant, si l'on adapte l'algorithme de Floyd-Murchland au cas symétrique, le nombre d'opérations élémentaires exigées par ce dernier algorithme n'est plus que de l'ordre de n^3 (et

non plus $2n^3$, puisqu'il suffit de calculer c_{ij} pour la moitié seulement des paires de sommets, à cause de la symétrie).

Résultats expérimentaux

Le tableau 3 présente les résultats numériques que nous avons obtenus pour l'algorithme DS. Il se lit comme le tableau 1. La tendance de T/T_{floyd} à décroître quand n croît s'y manifeste clairement.

	n	NPROB	n_1	n_2	NMOD	T	T_{floyd}	T/T_{floyd}
réseau # 5	20	10	4	3	13	.0007	.03	2.3%
réseau # 6	35	10	9	11	48	.0023	.15	1.5%
réseau # 7	50	10	21	9	103	.0040	.44	0.9%
réseau # 8	70	10	25	16	204	.0080	1.19	0.7%

TABLEAU 3

2.2 Augmentation de la longueur d'une arête

Si $c_{jk} < d_{jk}$, C reste inchangée. On supposera donc que

$$0 < c_{jk} = d_{jk} < d'_{jk} \leq M.$$

Proposition 10:

$$\left.\begin{array}{l} c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \\ \underline{\text{et}}\ c_{i\ell} \neq c_{ik} + d_{kj} + c_{j\ell} \end{array}\right\} \Longrightarrow \text{il ne faut pas modifier } c_{i\ell}.$$

Preuve:

$$\left.\begin{array}{l} c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \\ \underline{\text{et}}\ \ c_{i\ell} \neq c_{ik} + d_{kj} + c_{j\ell} \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} \text{la plus courte chaîne entre } i \text{ et } \ell \\ \text{n'emprunte pas l'arête } (j,k) \end{array}\right.$$

$\Longrightarrow$ il ne faut pas modifier $c_{i\ell}$

Fin de la preuve.

Corollaire 10a: Pour identifier les paires (i,ℓ) susceptibles d'être modifiées, il suffit de considérer l'ensemble

$$E = \{(i,\ell) \mid c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}\} \ .$$

Preuve: Par la proposition 10, on a:

$$c_{i\ell} \text{ susceptible d'être modifié} \Longrightarrow \left\{\begin{array}{l} c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} \\ \underline{\text{ou}}\ \ c_{i\ell} = c_{ik} + d_{kj} + c_{j\ell} \end{array}\right.$$

Si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$, alors $(i,\ell) \in E$.

Si $c_{i\ell} = c_{ik} + d_{kj} + c_{j\ell}$, alors $c_{\ell i} = c_{\ell j} + d_{jk} + c_{ki}$

et $(\ell,i) \in E$.

Proposition 11:

$$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow \left\{\begin{array}{l} i \text{ n'a pas à être considéré comme premier élément} \\ \text{d'une paire } (i,\ell) \text{ susceptible d'être modifiée.} \end{array}\right.$$

Preuve:

$$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow \forall \ell\ ,\ c_{i\ell} < c_{ik} + c_{k\ell} < c_{ij} + d_{jk} + c_{k\ell}$$

et par le corollaire 10a, on n'a pas à considérer i .

Proposition 12:

$$c_{j\ell} \neq d_{jk} + c_{k\ell} \Longrightarrow \begin{cases} \ell \text{ n'a pas à être considéré comme deuxième élément} \\ \text{d'une paire } (i,\ell) \text{ susceptible d'être modifiée.} \end{cases}$$

Preuve: $c_{i\ell} < c_{ij} + c_{j\ell} < c_{ij} + d_{jk} + c_{k\ell}$.

Le corollaire 10a et les propositions 11 et 12 mènent directement à l'algorithme AS (augmentation, symétrique).

L'algorithme AS

(Si $c_{jk} < d_{jk}$, C reste inchangée.)

as1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

as2) $\forall\, i \in N$, si $c_{ik} = c_{ij} + d_{jk}$, $N_1 := N_1 \cup \{i\}$.

as3) $\forall\, \ell \in N$, si $c_{j\ell} = d_{jk} + c_{k\ell}$, $N_2 := N_2 \cup \{\ell\}$.

as4) $\forall\, (i,\ell) \in N_1 \times N_2$, si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$,

on conserve la paire (i,ℓ) et $c_{i\ell} := d_{i\ell}$

$c_{\ell i} := d_{i\ell}$.

as5) Appliquer l'algorithme de Floyd-Murchland à toutes les paires (i,ℓ) conservées à l'étape as4, i.e.

<u>pour</u> m := 1 <u>jusqu'à</u> n

et pour chaque paire (i,ℓ) conservée

<u>si</u> $c_{im} + c_{m\ell} < c_{i\ell}$ <u>alors</u> $c_{i\ell} := c_{im} + c_{m\ell}$

et $c_{\ell i} := c_{i\ell}$.

Boyce *et al.* (1973, page 524) ont proposé un algorithme, qu'ils appellent "link deletion", pour accomplir le même travail que l'algorithme AS. Hormis le fait qu'il comporte deux légères erreurs qui pourraient bien n'être que des erreurs typographiques*, leur algorithme diffère essentiellement du nôtre en ce qu'il ne comporte pas l'étape as4. Or cette étape permet de réduire le temps de calcul, ainsi que le montre l'expérimentation (cf. tableau 4).

<u>Nombre d'opérations élémentaires</u>

Une simple comparaison entre l'algorithme A et l'algorithme AS nous indique qu'ils comportent exactement le même nombre d'opérations élémentaires, soit $2(2n + n_1 n_2 + n_{12} n)$ additions et comparaisons.

*<u>Note</u>: à l'étape 1 de l'algorithme "link deletion", il faudrait lire "$c_{jk} < d_{jk}$" au lieu de "$c_{jk} \leq d_{jk}$", et à l'étape 2(a), il faudrait lire "$c_{ik} = c_{ij} + d_{jk}$" et non pas "$c_{ij} = c_{ij} + d_{jk}$".

Résultats expérimentaux

Le tableau 4 présente les résultats numériques que nous avons recueillis avec l'algorithme AS. Il s'interprète exactement comme le tableau 2. On y observera encore une fois que le rapport T/T_{floyd} décroît quand n croît.

	n	NPROB	n_1	n_2	n_{12}	NMOD	T	T'	T_{floyd}	T/T_{floyd}
réseau # 5	22	10	3	5	13	13	.004	.005	.03	13%
réseau # 6	35	10	8	12	47	45	.019	.035	.15	13%
réseau # 7	50	10	30	5	81	79	.045	.077	.44	10%
réseau # 8	70	10	34	5	63	62	.049	.102	1.19	4%

TABLEAU 4

Références

Berge, C. (1967)
THEORIE DES GRAPHES ET SES APPLICATIONS
Dunod, Paris, 1967.

Boyce, D.E., Farhi, A. et Weischedel, R. (1973)
OPTIMAL NETWORK PROBLEM: A BRANCH-AND-BOUND ALGORITHM
Environment and Planning, vol. 5, pp. 519-533.

Dreyfus, S.E. (1969)
AN APPRAISAL OF SOME SHORTEST-PATH ALGORITHMS
Operations Research, vol. 7 no 3, pp. 395-412.

Floyd, R.W. (1962)
ALGORITHM 97: SHORTEST PATH
Comm. of ACM 5 (1962), p. 345.

Murchland, J.D. (1970)
A FIXED MATRIX METHOD FOR ALL SHORTEST DISTANCES IN A DIRECTED GRAPH AND FOR THE INVERSE PROBLEM
Thèse de doctorat, Université de Karlsruhe, 59 p.

Staire, S. (1968)
SELECTING AN OPTIMAL TRAFFIC NETWORK
Journal of Transport Economics and Policy, vol. 2, pp. 218-231.